DOI:10.2298/CSIS130304073B

Model-based Integration of Constrained Search Spaces into Distributed Planning of Active Power Provision

Jörg Bremer[1] and Michael Sonnenschein[1]

Department of Computing Science, University of Oldenburg
26129 Oldenburg, Gemany
{Joerg.Bremer, Michael.Sonnenschein}@uni-oldenburg.de

Abstract. The current upheaval in the electricity sector demands distributed generation schemes that take into account individually configured energy units and new grid structures. At the same time, this change is heading for a paradigm shift in controlling these energy resources within the grid. Pro-active scheduling of active power within a (from a controlling perspective) loosely coupled group of distributed energy resources demands for distributed planning and optimization methods that take into account the individual feasible region in local search spaces modeled by surrogate models. We propose a method that uses support vector based black-box models for re-constructing feasible regions for automated, local solution repair during scheduling and combine it with a distributed greedy approach for finding an appropriate partition of a desired target schedule into operable schedules for each participating energy unit.

Keywords: constrained optimization, support vector machines, smart grid, decoder-based optimization.

1. Introduction

In order to allow for a transition of the current central market and network structure of today's electricity grid to a decentralized smart grid, an efficient management of numerous distributed energy resources (DER) will become more and more indispensable. Integrating a continuously rising number of renewable resources means controlling individually configured and rather small devices in order to cope with stochastic feed-in effects. At the same time, more and more electricity is generated by wind energy converters and photovoltaic panels. A forecast on this generation highly depends on only fairly foreseen weather conditions and thus puts an additional challenge on planning the electricity provision and calls for additional operating reserve.

Within an electricity grid, electricity generation and consumption have to be balanced at every moment in time for physical reasons. In the past, only the generated part used to be planned according to anticipated consumption. Only few and large power plants had to be taken into account. As in future a larger share of the generation becomes hardly controllable (e. g. wind energy conversion or photovoltaics), parts of the demand have to be integrated into the planning process.

We consider in general producers that are supposed to pool together with likewise distributed electricity consumers and prosumers (like batteries) in order to jointly gain more degrees of freedom in choosing load schedules. In this way, they become a single controllable entity with sufficient market power. In order to manage such a pool of DERs in a self-organized way, the following distributed optimization problem has to be frequently solved: A partition of a demanded (by market) aggregate schedule has to be determined in order to fairly distribute the load among all participating DERs. Optimality usually refers to local (individual cost) as well as to global (e.g. environmental impact) objectives in addition to the main goal: Resemble the wanted overall load schedule as close as possible.

In order to choose an appropriate schedule for each participating DER, an optimization algorithm must know from each DER, which schedules are actually operable and which are not. Depending on the type of DER, different constraints restrict possible operations. The information about individual local feasibility of schedules has to be spread appropriately in (distributed) optimization scenarios, in order to evaluate objectives globally in distributed search spaces. For this purpose, meta-models of constrained spaces of operable schedules have been shown indispensable for efficient communication [10]. Such models can be seen as black-box representations of the feasible region of an optimization problem related to scheduling scenarios. Such models are also used for efficiently evaluating constraints during the optimization procedure for cases where determining the feasible region has comparably high computational costs.

Real world problems like this scheduling problem often face nonlinear and/ or combined constraints. The set of constraints defines the shape of a region within the search space (the hypercube defined by operation parameter limits) that contains all feasible solutions. This region is called feasible region and might be arbitrary shaped or even be discontinuous. It is this region that defines feasibility and that has to be modeled. Such a surrogate model then allows distinguishing operable and not operable schedules when integrated into the optimization process.

At the same time, support vector machines and related approaches have been shown to have excellent performance when trained as classifiers for multiple purposes, especially real world problems. As a use case related to describing the region where some given data resides in, Tax and Duin developed the support vector domain description (SVDD) as a one-class support vector classification approach that is capable of learning the region that is defined by some given training data [43] and that has therefore been harnessed for example by [7] as a model for the feasible region in the smart grid domain.

What we will now add is a new approach for integrating constraints that are modeled by a support vector classifier into distributed optimization in a way, that allows for an efficient navigation within the feasible region. The basic idea is to construct a mapping from the whole, unconstrained domain of the problem (the hypercube) to the feasible region to be able to automatically repair an infeasible solution during optimization. In this way, the problem is transferred into an unconstrained one by mapping any arbitrary solution onto a nearby feasible one. What we will need for constructing this mapping is the set of support vectors together with the associated weights that make up the black-box model.

The rest of the paper is organized as follows: We start with a discussion of related approaches and the background of the optimization problem that is considered throughout this paper. We describe a model of individual search spaces of arbitrary energy resources based on a support vector approach and the behaviour model based method for learning it. Then, we define the mapping function that is used as solution repair mechanism within our greedy algorithm for scheduling. Finally, we conclude with results from several simulation runs and with a discussion of a possible extension to an asynchronous execution of the optimization.

2. Related work and problem background

Within the framework of today's (centralized) operation planning for power stations, different heuristics are already harnessed. Examples from the research sector are for instance shown in [28] or in [47]. The task of (short-term) scheduling of different generators is also known as unit commitment problem and assigns (in its classical interpretation) discrete-time-varying production levels to generators for a given planning horizon [36]. It is known to be an NP-hard problem [17]. Determining an exact global optimum is, in any case, not possible until ex post due to uncertainties and forecast errors. In practice the software package BoFIT is often used, harnessing a mixed integer model with operational constraints as an integral part of the implementation of the model [14]. This fact makes it hard to exchange operational constraints in case of a changed setting (e.g. a new composition of energy resources) of the whole generation system.

Coordinating a pool of distributed generators and consumers with the intent to provide certain aggregated load schedules for active power has some objective similarities to controlling a virtual power plant (VPP) [46, 27]. Within the smart grid domain the volatile character of such a group has additionally to be taken into account. On an abstract level, approaches to control groups of distributed devices can be roughly divided into centralized and distributed scheduling algorithms.

Centralized approaches have long time dominated the discussion [46], not least because a generator may achieve slightly greater benefit if optimization is done from a global, omniscient perspective [16]. Centralized methods are discussed in the context of static pools of DERs with drawbacks and restrictions regarding scalability and particularly flexibility.

Recently, distributed approaches gained more and more importance. Different works proposed hierarchical and decentralized architectures based on multi-agent systems and market based computing [21, 22]. Newer approaches try to establish self-organization between actors within the grid [20, 29, 38].

Especially for optimization approaches in smart grid scenarios, black-box models for encoding the feasible region with the set of operable schedules have been developed [10]. The encoding of a schedule's individual cost may also be easily embedded into such model [8].

This quite new support vector approach uses support vector meta-models for black-box optimization scenarios with no explicitly given constraint boundaries. In general, various classification or regression methods could be harnessed for creating such models for the boundary [23], but not all of them allow for a good integration into optimization. In general, there are three main reasons for using such a model-based approach:

1. Substituting computational costs for evaluating the constraints by a comparatively easy check on feasibility through the model.
2. Efficient communication in distributed environments due to the small footprint of the model.
3. Unification of access to the information on feasibility.

Besides, the smart grid domain serves also as an example for scenarios with (at least partly) unknown functional relationships of the constraints. The feasible region can sometimes only be derived with lacking full knowledge on hidden variables or intrinsic relations that determine the operability of an electric device and therewith the feasible region. The authors therefore have their model learned by a support vector data description approach from a set of operable (feasible) examples.

In a related approach for another use-case, [4] used a two-class SVM for learning operation point and bias of a line in a power grid for easier determining an optimal way back to stable grid conditions in case of a failure.

Surrogate modeling (also known as surface or meta modeling) is often used for replacing expensive and time consuming evaluations of simulations by a model that mimics system behavior on a local or global level [11] with a minimum of known samples from the original (simulation) model. For this reason, active sampling is often applied, i.e. the sample that makes up the model is continuously adapted and iteratively improved as analysis or optimization evolves. Several different techniques for surrogate modeling have been developed. Commonly used examples are: Datascape, Kriging, first and second order regression, response surfaces, or artificial neural networks [15, 31]. Using surrogate modeling like surface modeling in optimization, usually, all constraints are directly known in contrast to the function that is to be optimized along [32]. In our problem, we want to abstract from certain given constraint formulations and do (at optimization time) not have access to the individual simulations of the distributed resources.

For our optimization problem, we need a model that serves as a stencil for the set of (technically) operable schedules for a DER. We regard the respective simulation model as a characteristic function that is able to indicate whether an arbitrary, given schedule is operable by the DER or not. We want to build a model that is able to guide an optimization algorithm towards feasible solutions. Thus, we need a model that is able to capture the characteristic function that indicates operability of schedules. As we do not have access to the simulation model at optimization time anymore, we have to build the model completely in advance and are not able to come back to iteratively improving modeling techniques like active learning. A support vector model can do this and an important advantage of such a model for our use case is the internal representation that can easily and directly be used for further calculations.

In this paper, we will consider the following optimization problem for a given group (consumers, producers and/ or prosumers) of DERs: A schedule for a given future time horizon is requested (e.g. via an electricity market mechanism) and is supposed to be jointly operated by the group. We developed a general method for arbitrarily composed groups of different types of DER. As the method will abstract from precise modeling of each DER as well as from constraint modeling, it is suitable especially for groups that dynamically reformate frequently in a self-organizing system. For an (not necessarily known in advance) group, a partition has to be found. Thus, we do not assume a certain size or composition of different DERs. A partition of the requested target schedule has to be determined in order to fairly distribute the load among all participants.

With the term load, we denote the mean electrical active power that is produced or consumed by a DER within a certain time interval (today: usually 15 minutes). A schedule then is a vector that determines the loads for a given number of subsequent time intervals. This definition is equivalent to defining a schedule by using the respective amounts of energy produced or consumed within a time interval.

For the sake of simplicity, we will consider optimality as a close as possible adaption of the aggregated schedule (sum of individual loads) to the requested one during this paper. Optimality usually refers to additional local (individual cost) as well as to global (e.g. environmental impact) objectives. As we present a general approach that is independent of a certain optimization objective, we have chosen this simple version for demonstration purposes. When coming into operation, one would use a more sophisticated objective; and usually a many-objective approach that takes into account local (individual user-specific) optimality as well as additional global objectives like minimizing the coincidence factor [1].

When determining an optimal partition of the target schedule for load distribution, exactly one alternative schedule is taken from each DER's search space of individual operable schedules in order to assemble the desired aggregate schedule. Therefore, the optimization problem is to find any combination of schedules (one from each DER with $\mathcal{F}_i$ as the set of possible choices, i.e. the individual feasible region) that resembles the target schedules s_{target} as close as possible, i.e. minimize the Euclidean distance ($\|\cdot\|$) between aggregated and target schedule:

$$\|\sum_i x_i - s_{target}\| \rightarrow \min, \tag{1}$$

such that each schedule is taken from the respective feasible region $\mathcal{F}_i$ of operable schedules

$$x_i \in \mathcal{F}_i$$

of unit i. The term unit in this context denotes a single DER or an aggregation of commonly controlled energy resources, e.g. the set of all controllable appliances in a household that (from an outside position) can be seen as a single controllable unit. The individual schedules as well as the wanted target schedule are each given as a vector $x_i, s_{target} \in \mathbb{R}^d$ that represents with its elements the mean active power during the respective time period (usually but not necessarily a 15-minute interval). Usually the problem consist of additional objective functions and results in an many objective problem. For demonstration purposes, we will stick with the single objective problem throughout the rest of the paper.

Moreover, the choice of using the Euclidean distance as metric would have to be reconsidered according to the given problem at hand as it is known that the distance of two arbitrarily chosen points tends to approach 1 with growing dimensionality. The same problem holds true when learning a model. For kernel based approaches, [13] proposes methods to overcome this problem. For planning periods of one day with 96 intervals of 15-minutes, the choice will be sufficient for the case of the objective function. As an alternative distance, for example [18] uses the L_1-distance. Depending on special objectives that one wants to achieve (e.g. minimizing surplus production), maybe also metrics like excess supply, are appropriate. An overview on methods to assess the match between schedules can for example be found in [5].

The following section will explain our approach for solving this optimization problem with individually acting DER as part of a coalition of DERs in a distributed approach. Although the problem is defined on a group of DERs, we will first have to look at a single unit and on how to model its abilities before putting together the models from each DER into a jointly solved optimization approach for the multi DER problem.

If such a group of DERs consists of individually operated units, we first have to determine which set of alternative schedules each unit has to offer for the afterwards optimization step. A schedule for d time intervals will be geometrically interpreted as a point in $\mathbb{R}^d$. The model that we present will be applicable to arbitrary types of DER but we will restrict our explanations to the example of a co-generation plant.

3. The model-based optimization approach

3.1. The feasible region

Each DER has to serve the purpose it has been built for and this purpose may usually be achieved in different alternative ways. For example, it is the main purpose of a μCHP (combined heat and power generator) to deliver enough heat for varying heat demands in a household at every moment in time. Nevertheless, heat usage is usually decoupled from heat production by using a thermal buffer store. Thus, different production profiles may be used for generating the heat. This leads, in turn, to different respective electric load profiles that may be offered as alternatives to a scheduling controller.

Each DER offers a set of operable schedules for a given (future) time horizon. We see a schedule as a data vector $x \in \mathbb{R}^d$, with the number of periods d. For each period the i-th element of x describes the respective amount of electric energy produced or consumed in this period or respectively the mean active power output or input during this period.

The term operable in this context means that such a schedule might be operated by the DER without violating any technical constraint. Moreover, we consider additional non-technical constraints that may restrict the possible operations of a DER. Constraints can be distinguished into hard (usually technically rooted) and soft constraints (often economically or ecologically rooted or subject to personal preferences).
Examples for hard constraints are:

- Minimum and/or maximum power input/output
- Integrated amount of energy produced over the given time frame
- Restrictions on thermal buffer storage
- Achieve intended purpose

Examples for soft constraints are:

- Costs (e.g. fuel costs) for operating a certain schedule
- Environmental performance
- Personal preferences (e.g. noise pollution in the evening)

These constraints can be interpreted geometrically. Without any constraint, the whole hypercube $[0, 1]^d$ (active power between 0 and 100%) would be a model for the region of feasible schedules. With each constraint, a different part (region) of the hypercube falls off the feasible region, because the respective schedules are not operable due to the constraint. Only the finally remaining region (hypercube minus superposition of all regions prohibited by constraints) is the feasible region of the DER. Only from this region, schedules might be taken during optimization.

It has been shown in [10] that the feasible region of operable schedules of a DER is not necessarily a convex polytope nor a single and connected region. For this reason, concavity and clusters have to be taken into account, too. Such considerations have led to black-box models based on machine learning approaches that may

- capture the topological traits of the feasible region as a compact description of the set of all operable schedules.
- be easily communicated as a standardized description within distributed optimization scenarios.
- ease the calculation of the feasibility of a solution during optimization.

Before we discuss a new way of integrating this model directly into optimization, we will briefly discuss the basic idea of the model approach.

3.2. Support vector black-box model for constraints

We will describe the black-box model for the set of feasible schedules for a DER as it has been developed in [10] based on a one-class support vector data description (SVDD) [44]. The goal of building such a model is to learn the feasible region of the schedules of a DER by learning the enclosing boundary around the set of operable schedules. This task is achieved by determining a mapping $\Phi : \mathcal{X} \subset \mathbb{R}^d \to \mathcal{H}, x \mapsto \Phi(x)$ such that all data from a given region $\mathcal{X}$ is mapped to a minimal hypersphere in some high- or indefinite-dimensional space $\mathcal{H}$. Originally, the use case was to use this model as a classifier that allows for distinguishing operable and not operable schedules during optimization without explicit knowledge about the constraints that restrict the operations of the DER.

The minimal sphere with radius R and center a in $\mathcal{H}$ that encloses $\{\Phi(x_i)\}_N$ can be derived from

$$\|\Phi(x_i) - a\|^2 \leq R^2 + \xi_i \quad \forall i \tag{2}$$

with $\|\cdot\|$ denoting the Euclidean norm and incorporating slack variables $\xi_i \geq 0$ that introduce soft constraints for sphere determination.

After introducing Lagrangian multipliers and further relaxing to the Wolfe dual form, the well known Mercer's theorem [39] may be harnessed for calculating dot products in $\mathcal{H}$ by means of a Mercer kernel in data space:

$$\Phi(x_i) \cdot \Phi(x_j) = k(x_i, x_j). \tag{3}$$

In order to gain a more smooth adaption, it is known [3] to be advantageous to use a Gaussian kernel:

$$k_{\mathcal{G}}(x_i, x_j) = e^{-\frac{1}{2\sigma^2}\|x_i - x_j\|^2} \tag{4}$$

instead of for instance polynomial kernels.

Putting it all together, the equation that has to be maximized in order to determine the desired sphere is:

$$W(\beta) = \sum_i k(x_i, x_i)\beta_i - \sum_{i,j} \beta_i \beta_j k(x_i, x_j). \tag{5}$$

Maximizing (5) is a problem of quadratic programming (QP) [42], which is known to be of cubic computational complexity $\mathcal{O}(N^3)$ with sample size N [12]. For this reason, the adoption of a technique called sequential minimal optimization [37] (SMO) is used for solving Eq. (5). SMO breaks up the large QP problem for SVM training into a series of smallest possible subproblems which can be solved analytically. In future, if real-time constraints might be involved, SVM training may be done incrementally with an online learning algorithm [25].

In this way, working on the data points one by one, it becomes possible to break the process with the so far reached result if a deadline for answering is approaching.

The result (weight vector β) represents the center a of the sphere in terms of an expansion into $\mathcal{H}$:

$$a = \sum_i \beta_i \Phi(x_i). \tag{6}$$

The distance R of the image of an arbitrary point $x \in \mathbb{R}^d$ from $a \in \mathcal{H}$ can be calculated in $\mathbb{R}^d$ by:

$$R^2(x) = 1 - 2\sum_i \beta_i k_{\mathcal{G}}(x_i, x) + \sum_{i,j} \beta_i \beta_j k_{\mathcal{G}}(x_i, x_j). \tag{7}$$

Finally, the radius $R_{\mathcal{S}}$ of the sphere $\mathcal{S}$ is determined by the distance to a of an arbitrary support vector as these are mapped right onto the surface. Thus the original feasible region is now modeled as

$$\{x \in \mathbb{R}^d | R(x) \leq R_{\mathcal{S}}\}. \tag{8}$$

3.3. The model of the search space

The model of the feasible region of an arbitrary DER consists of the set s of support vectors and respective weights from β as this is all that is needed for reconstructing the boundary that encloses the feasible region. From β only the non zero components for the support vectors are necessary. We denote this reduced weight vector with w. Formally, the model consists of:

- Set of support vectors (example schedules) $SV = \{x_i \in \mathcal{X} \mid \beta_i \neq 0\}$
- Associated weights: $w = (\beta_1, \ldots, \beta_n) \quad \forall \beta \neq 0$
- Some additional parameters: e.g. max. power, cost factor, ...
- Decision function: $R^2(x) = 1 - 2\sum_i w_i k_{\mathcal{G}}(s_i, x) + \sum_{i,j} w_i w_j k_{\mathcal{G}}(s_i, s_j)$ for deciding on feasibility of a given schedule on for comparing two given schedules and deciding on which is nearer to feasibility.
- Feasible region : $\{x | R(x) \leq R_{\mathcal{S}}\}$

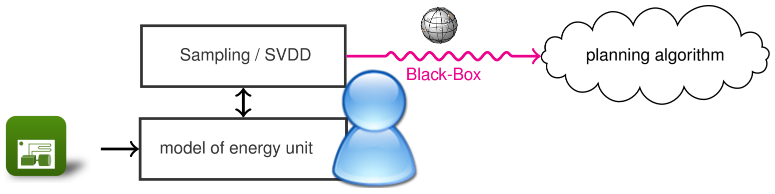


Fig. 1. Integrating the different models into an agent-based energy planning process.

The model-based planning approach that is described later, is currently integrated into a multi-agent simulation of large, future smart grid scenarios. Figure 2 shows the interplay of the involved models from an architectural point of view. As we aim at a system that enables self-organized and market-based control of distributed energy production [34], we go for an agent-based system. The whole system [33, 41, 2] will comprise several types of agents for different tasks. Among them are: market agents for different markets (real power, ancillary services, operating reserve, etc.), agents representing a DER during coalition forming and negotiation at market, or grid agents in charge of checking and assuring grid compatibility. For the rest of the paper, we will focus on the type of agent in charge of controlling a DER. In this way, the search space model represents the feasible part of the possible future behaviour of an energy unit as it is learned from a behaviour model that simulates actual device. On an agent interaction level, only the search space model is used as a surrogate for the unit's behaviour.

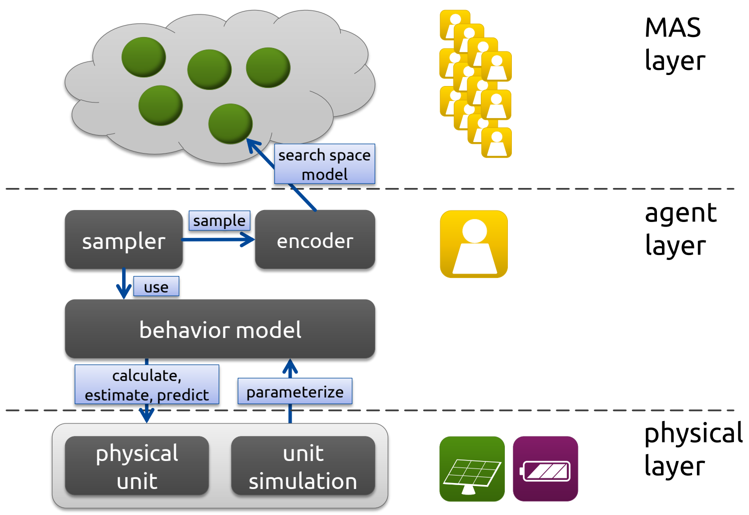


Fig. 2. Using the model inside the multi-agent system hierarchy.

Prior to determining the describing support vectors, the set of training data points $\mathcal{X}$ itself has to be determined. We do this by means of a mathematical model of a DER that can be parameterised with the current (measured) state of the device. This model must be at least able to verify (compliant with all constraints) or falsify (can not be operated) given schedules.

In many cases, it is far too complex to enumerate all possible schedules and to check them against the model. An example with 100 discrete power levels would lead to 100^{96} schedules that would have to be checked for a conventional day-ahead (horizon with 96 periods) scenario. For this reason, one would draw a comparatively small random sample from the set of all schedules.

However, feasible schedules reside in a region that is extremely small compared with $[0, p_{max}]^d$. This fact applies in particular to high dimensional schedules. If, for example, for each period one third of the alternatives is prohibited by constraints, then the ratio of feasible solutions to all solutions amounts to $(\frac{2}{3})^{96} \approx 1.25 \times 10^{-17}$ for the general case of $d = 96$ periods.

Considering load profiles for a whole day, an investigation of our simulation models has shown a proportion of valid load profiles below 10^{-23}. For this reason, it is impossible just to draw random load profiles and check their validity with the model in acceptable time. Hence, we currently draw samples as an successive drawing of period wise guessing.

1. Guess a random power level for just one period.
2. Validate this 1-dimensional schedule with the help of the DER model.
3. If it is valid: Simulate the follow-up state of the DER, re-parameterize the model and goto 1. for determination of the next period.

This approach has the advantage of leading far more likely to guesses of valid schedules. The probability $\mathcal{P}$ for guessing a valid schedule for a single period is already rather high. Allowing for multiple guessing (with number of tries n) results in the even higher probability

$$\mathcal{P}_{(n)} = \sum_{i=1}^{n} B(i|\mathcal{P}, n) = \sum_{i=1}^{n} \binom{n}{i} \mathcal{P}^i (1 - \mathcal{P})^{n-i}, \tag{9}$$

where $\mathcal{P}_{(n)}$ is the probability for at least one successful guess within n tries. Successive guessing then results in an overall probability for successfully guessing a complete schedule of d periods of

$$\mathcal{P}^d_{(n)} = \left(\sum_{i=1}^{n} B(i|\mathcal{P}, n) \right)^d . \tag{10}$$

As an example, let the probability of correctly guessing a valid power load level for a single period be 0.05. Allowing for 100 guesses for each period, then the overall probability for guessing a schedule of 96 periods correctly is still $\mathcal{P}^{96}_{(100)} = 0.5655$, which is sufficiently high in contrast to 10^{-23} (as had been estimated for some of our models).

A major drawback of this approach is that in this way we get a set of schedules that is dominated by the first periods. That means, schedules do not have equal probability for being chosen. This disadvantage is currently deferred for two reasons:

1. We are not primarily interested in statistical properties of the sample or the underlying density. Instead, we want to sound the geometric region where valid schedules reside in.

2. Simulation runs have shown that the principle structures of the scopes of action are nevertheless revealed with this method - as long as merely geometric properties are considered.

An interactive method for a sampler that uses a simulation model that implements the behaviour of an energy unit (c.f. figure 2) is shown in figure 3.

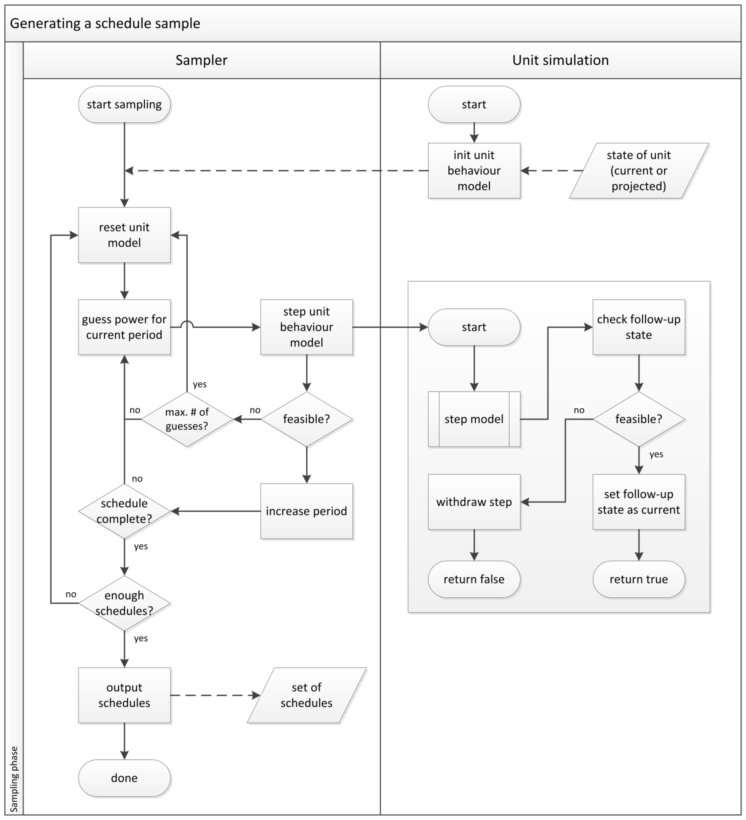


Fig. 3. Sampling process for the training sample prior to learning the search space model.

This process makes use of equation (10) and guesses for a given number of tries a short schedule (usually one period) for the near future until one is found that is actually operable. The behaviour model checks the validity and conditionally calculates the follow-up state resulting from operating the guessed schedule. This step is repeated until a schedule of sufficient length is found. In order to use an appropriate likelihood distribution for our guesses, we a priori make a kernel density estimation of the distribution of operable parts of the schedule [35].

3.4. Constructing solutions from the model

We will now show a way to use this model in a more sophisticated way than as a mere black-box model. We are interested in having a means of finding a nearby feasible schedule next to an arbitrary given schedule. For this purpose, we will harness a function that maps the d-dimensional unit hypercube (representing arbitrary schedules in a scaled scenario) onto the feasible region. In this way, any (in-)feasible schedule will be converted into a feasible one. The construction of this mapping is described in this section.

For our use case, we need a procedure that generates a nearby and feasible solution from any given (likely not feasible) schedule. Nearby in this context means that the distance in solution space between given and near feasible solution is small. This task can be achieved by constructing a mapping that maps every infeasible point from input space into or onto the feasible region. We have tested both approaches. Here, we describe the more general case of mapping into the feasible region that includes the specialized case. In general, this mapping can also be used for transforming the whole optimization problem into an unconstrained one.

Let $\mathcal{F}$ denote the feasible region within the domain of some given optimization problem bounded by an associated set of constraints. It is known, that preprocessing the data by scaling it to $[0,1]^d$ leads to better adaption [19]. Considering optimization problems in the energy sector, rescaling of the domain to $[0,1]^d$ leads to some advantages [7]. For this reason, we here consider scaled domains, too, and denote with $\mathcal{F}_{[0,1]}$ the likewise scaled region of feasible solutions. We want to construct a mapping

$$\begin{aligned} \gamma : [0,1]^d &\rightarrow \mathcal{F}_{[0,1]} \subseteq [0,1]^d \\ x &\mapsto \gamma(x) \end{aligned} \tag{11}$$

that is able to map the unit hypercube $[0,1]^d$ onto the d-dimensional region of feasible solutions that is bounded by a set of arbitrary (maybe nonlinear) constraints. But, instead of directly mapping to $\mathcal{F}_{[0,1]}$ we will go through the kernel space as shown in the following commutative diagram (12).

$$\begin{tikzcd} x \in [0,1]^d \arrow[r, "\hat{\Phi}_\ell"] \arrow[d, "\gamma"'] & \hat{\Psi}_x \in \mathcal{H}^{(\ell)} \arrow[d, "\Gamma_a"] \\ x^* \in \mathcal{F}_{[0,1]} \subseteq [0,1]^d & \tilde{\Psi}_x \in \mathcal{H}^{(\ell)} \arrow[l, "\Phi_\ell^1"] \end{tikzcd} \tag{12}$$

We start with an arbitrary point $x \in [0,1]^d$ from the unconstrained d-dimensional hypercube and map it to an ℓ-dimensional manifold in kernel space that is spanned by the images of the support vectors $s_1 \ldots s_\ell$. After drawing this mapped point towards the sphere in order to pull it into the image of the feasible region, we look for the pre-image of the moved image to get a point from $\mathcal{F}_{[0,1]}$. Thus, we achieve the wanted mapping as a composition of three functions:

$$\gamma = \Phi_\ell^1 \circ \Gamma_a \circ \hat{\Phi}_\ell. \tag{13}$$

We will now look at each step in more detail.

Mapping x to the support vector induced subspace $\mathcal{H}^{(\ell)}$ with an empirical kernel map Let

$$\begin{aligned} \Phi_\ell : \mathbb{R}^d &\to \mathbb{R}^\ell, \\ x &\mapsto k(.,x)|_{\{s1,\dots,s_\ell\}} \\ &= (k(s_1,x),\dots,k(s_\ell,x)) \end{aligned} \tag{14}$$

be the empirical kernel map w.r.t. the set of support vectors $\{s_1,\dots,s_\ell\}$. If Φ_ℓ is modified to

$$\hat{\Phi}_\ell : x \mapsto K^{-\frac{1}{2}}(k(s_1,x),\dots,k(s_\ell,x)) \tag{15}$$

with $K_{ij} = k(s_i,s_j)$ being the kernel Gram Matrix, then function Eq. 15 maps points x,y from input space to $\mathbb{R}^\ell$, such that $k(x,y) = \hat{\Phi}_\ell(x) \cdot \hat{\Phi}_\ell(y)$ (cf. [39]).

With $\hat{\Phi}_\ell$ we are able to map arbitrary points from $[0,1]^d$ to some ℓ-dimensional space $\mathcal{H}^{(\ell)}$ that contains a projection of the sphere. Again, points from $\mathcal{F}_{[0,1]}$ are mapped into or onto the projected sphere, outside points go outside the sphere and must be moved in $\mathcal{H}^{(\ell)}$ towards the center in the next step in order to draw them into the image of the feasible region.

Re-adjustment in kernel space In general, in kernel space $\mathcal{H}$ the image of the region is represented as a hypersphere $\mathcal{S}$ with center a and radius $R_\mathcal{S}$ (Eq. 7). Points outside this hypersphere are not images of points from $\mathcal{X}$, i.e. in our case, points from $\mathcal{F}_{[0,1]}$ are mapped (by Φ) into the sphere or onto its surface (support vectors), points from outside $\mathcal{F}_{[0,1]}$ are mapped outside the sphere. Actually, using a Gaussian kernel, Φ maps each point into an at most n-dimensional manifold (with sample size n) embedded into infinite dimensional $\mathcal{H}$. In principle, the same holds true for a lower dimensional embedding spanned by ℓ mapped support vectors and the ℓ-dimensional projection of the hypersphere therein.

We now want to pull points from outside the feasible region into that region. As we do have rather a description of the image of the region, we draw images of outside points into the image of the region, i.e. into the hypersphere; precisely into its ℓ-dimensional projection. For this purpose we use

$$\tilde{\Psi}_x = \Gamma_a(\hat{\Psi}_x) = \hat{\Psi}_x + \mu \cdot (a - \hat{\Psi}_x) \cdot \frac{R_x - R_\mathcal{S}}{R_x} \tag{16}$$

to transform the image $\hat{\Psi}_x$ produced in step 1) into $\tilde{\Psi}_x \in \hat{\Phi}_\ell(\mathcal{F}_{[0,1]})$ by drawing $\hat{\Psi}_x$ into the sphere. Alternatively, the simpler version

$$\tilde{\Psi}_x = a + \frac{(\hat{\Psi}_x - a) \cdot R_\mathcal{S}}{R_x} \tag{17}$$

may be used for drawing $\hat{\Psi}_x$ just onto the sphere but with the advantage of not having to estimate parameter $\mu \in [1, R_x]$. Parameter μ allows us to control how far a point is drawn into the sphere ($\mu = 1$ is equivalent to Eq. (17), $\mu = R_x$ draws each point onto the center). In this way, each image is re-adjusted proportional to the original distance from the sphere and drawn into the direction of the center.

Points from the interior are also moved under mapping gamma in order to compensate for additional points coming from the exterior. In this way, the whole unit hypercube is literally squeezed to the form of the feasible region without a too large increasing of the density at the boundary. Though, if the feasible region is very small compared with the hypercube, density at the boundary increases (depending on the choice of μ). On the other hand, the likelihood of an optimum being at the boundary increases likewise. So, this might be a desired effect.

After this procedure we have $\tilde{\Psi}_x$ which is the image of a point from $\mathcal{F}_{[0,1]}$ in terms of a modified weight vector $\tilde{w}^{\Gamma_a}$.

Finding an approximate pre-image As a last step, we will have to find the pre-image of $\tilde{\Psi}_x$ in order to finally get the wanted mapping. A major problem in determining the pre-image of a point from kernel space is that not every point from the span of Φ is the image of a mapped data point [39]. As we use a Gaussian kernel, actually none of our points from kernel space can be related to an exact pre-image except for trivial expansions with only one term [24]. For this reason, we will look for an approximate pre-image whose image lies closest to the given image using an iterative procedure after [30]. In our case (Gaussian kernel), we iterate x^* to find the point closest to the pre-image and define approximation Φ_ℓ^1 by equation

$$x^*_{n+1} = \frac{\sum_{i=1}^{\ell} (\tilde{w}_i^{\Gamma_a} e^{-\|s_i - x^*_n\|^2/2\sigma^2} s_i)}{\sum_{i=1}^{\ell} (\tilde{w}_i^{\Gamma_a} e^{-\|s_i - x^*_n\|^2/2\sigma^2})}. \qquad (18)$$

As an initial guess for x^* we take the original point x and iterate it towards $\mathcal{F}_{[0,1]}$. As this procedure is sensitive to the choice of the starting point, it is important to have x as a fixed starting point in order to ensure determinism of the algorithm. This is an essential requirement at least for integration into evolutionary algorithms since the same schedule has to be mapped several times, e.g. during search and again after optimization when the best found solution configuration is finally converted into the solution. Empirically, x has showed up to be a useful guess.

Eventually, we have achieved our goal to map an arbitrary point from $[0, 1]^d$ into the region of feasible solutions described merely by a given set of support vectors and associated weights: x^*_n is the sought after image under mapping γ of x that lies in $\mathcal{F}_{[0,1]}$.

This model and mapping may be used in different ways during optimization. Among these use cases are:

– Repair of infeasible solutions.

– Transformation of an constrained to an unconstrained optimization problem by mapping the whole search space into the feasible region.
– Computational easy classification of an solution's feasibility.
– Compact communication of large sets of operable schedules.

Here, we will harness the capability of repairing infeasible solutions for a distributed optimization approach.

3.5. The distributed greedy algorithm

With the above sketched preliminaries, we are now able to define our optimization algorithm. In order to pay attention to the ongoing decentralization of electricity grid control, it seems way more promising to design the optimization process distributed, too. In addition, the chances for success in finding an exact solution are rather low due to problem size, what makes a heuristic most suitable.

In this sense, we propose the following greedy algorithm for approximately solving optimization problem equation (1). In our optimization scenario, we assume one type of agent: one control agent for each a single energy resource with the following responsibilities/ capabilities:

– Simulating the underlying physical device in order to determine operable example schedules.
– Calculation of the support vector based black-box model.
– Calculation of mapping γ.
– Determining the schedule for the agent's own physical device that minimizes the overall loss.
– Participation in some joint optimization process.

The procedure for optimizing the aggregated schedule is now the one depicted in Fig. 4. Within a group of agents $\mathcal{A}$, one agent is randomly chosen to start the procedure. Here, we assume an agent to be in charge of controlling a DER and to participate in the distributed procedure of determining schedules for each DER such that the aggregated schedule best fits a given objective schedule. An elected initiator initializes the solution with all values to zero. Then, solution improvement begins. The agent adds up all schedules (known from the solution object) from all other agents. This is equivalent to subtracting one's own schedule from the aggregated solution. In a next step, the difference vector Δx of this sum to the desired target schedule is determined. This difference vector Δx represents the optimal schedule for the current agent in the following sense: if the agent would be able to deliver this schedule, the target could be reached exactly. Therefore, the agent now determines the nearest schedule to Δx that is actually operable by the device. This nearby schedule can be easily calculated with the help of the mapping γ that has been described in the previous section. Function γ maps an arbitrary schedule (in our case difference schedule Δx) into the region of feasible schedules and delivers the respective operable schedule that is close to Δx, because it uses the shortest trace to the feasible region to

```
$\mathcal{A} \leftarrow$ List of all agents
**if** is initiator **then**
  $S \leftarrow zeros(n, d)$
**else**
  $S \leftarrow$aggregated schedule
  $S_{new} \leftarrow \gamma(T - (S - S_a))$
  $S \leftarrow S - S_a + S_{new}$
  **if** no stop criterion met **then**
    choose random agent $A \in \mathcal{A}$
    send message with $S$ to $A$
  **else**
    publish solution $S$
  **end if**
**end if**
```

Fig. 4. Greedy algorithm (c.f. [9]) that each agent repeatedly executes for successive solution improvement starting from a zero solution S with S denoting the aggregated overall solution and S_a denoting the individual current contribution of the agent.

move a point. Please note that the distance between sum and target schedule that we minimize by this approach is the Euclidean distance L_2.

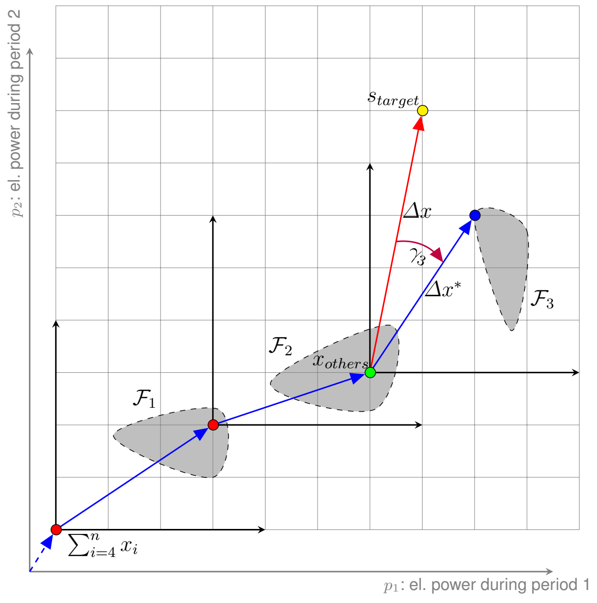
Figure 5 illustrates the approach with showing the situation at two successive iterations. The figure shows a 2-dimensional example problem. In the first step (fig. 5(a)) it is the turn of agent no. 3. Point x_{other} denotes the sum of current schedules of all other agents (from the perspective of agent 3). Vector Δx denotes the difference that is necessary to achieve the target schedule s_{traget} exactly. Because it is usually not necessarily the case that the difference Δx is feasible for the agent's energy unit, Δx is mapped to the feasible region by mapping γ resulting in the nearby schedule Δx^* that is feasible for agent 3. This schedule Δx^* is then taken as the best what agent 3 currently can do and is set as the agent's current schedule.

In the next step (fig. 5(b)) agent 2 becomes active. The sum of all other agents now is different because of the different perspective and due to new schedule of agent 3 from the previous step. Agent 2 does the same steps as previously agent 3 resulting in an updated schedule for agent 3 with a degradation ΔE of the overall error E. The figure also shows how individual schedules are moved to the respective feasible regions (grey areas) to ensure operability of the solution.

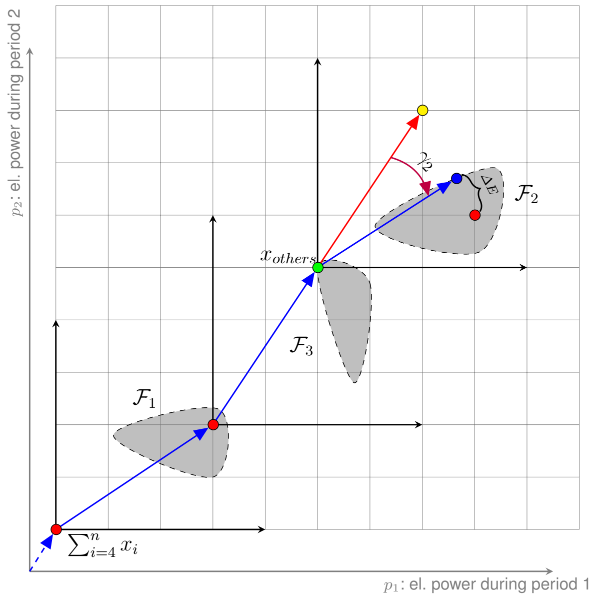
In this way, each DER chooses a schedule that is a compromise of being feasible (automatically ensured by mapping γ) and doing one's own best in bringing forth the overall solution towards the wanted adaption to the target schedule as much as possible each time when it is the respective agents turn.

As a stop criterion, we chose a maximum number of iterations at which the term iteration refers to one execution of the procedure in Fig. 4 by one agent.

Additional objectives could for example be integrated by having different cost indicators added as additional elements to the electrical schedule (in this way,



(a)



(b)

Fig. 5. Base principle of the greedy approach. Part (a) shows an optimization step with an active agent no. 3 doing an improvement by mapping the residual Δx onto his feasible region (grey area). Part (b) shows the follow-up step with an active agent no. 2.

combining schedule and evaluation criteria to a feature vector) and have the relation of schedule and evaluation criteria learned concurrently with the same approach as described here. As an example, this has been done with environmental criteria [6]. Each agent in the greedy optimization approach would then try to reach the electric active power target and a good value for each indicator at the same time, trying to reach a value of 0 for each criterion that has to be minimized and 1 else (provided that all criteria are likewise scaled to $[0, 1]$) by taking these additional targets into account when calculating Δx.

By one after another, the overall solution (the aggregated schedule) is successively improved. We have chosen to activate the agents in a random order, but a round robin approach may also do if each agent knows about his successor. In this way, the algorithm is distributed and sequential as only one agent has the token to work at a time. If the objective is to adapt to a given target schedule, the only information that has to be passed around (or made globally available) is the aggregated overall solution (as sum of all local solutions) and the desired target schedule. This is sufficient as each agent may remember his own local schedule that has been determined previously. All other information can be determined by local information.

Clearly, the actual optimization is distributed but sequential. But, the most time consuming part – namely learning the model for the computation of mapping γ – can be done in advance and fully parallel, what in turn allows for faster optimization afterwards without a need for considering constraints anymore.

Finally, we will discuss the ability of the whole algorithm to be run in parallel. In this case, two realizations are possible. First, the process as described above could be further parallelized by making the update calculation asynchronously run. In this way, an agent would first choose and trigger a successive agent and then calculate his own update. A problem here lies in the responsibility for detecting sufficient convergence.

Another approach would be a realization with a central instance that holds the current solution and provides it to all agents. Then, each agent could asynchronously and without any further trigger repeatedly query for the current solution and update the own solution part. Storing the new overall solution should of course be an atomic operation. In this case, the central instance (e.g. the coalition leader) that holds the current solution would also be in charge of deciding on convergence and on bringing the process to a hold.

4. Simulation results

So far, we have tested our approach with several simulated energy resources in different groupings. Among them are: co-generation devices with thermal buffer store and a simulated residential thermal energy demand as well as simulated controllable cooling devices. We will here focus on results from CHP generation. All simulations have been done with power scaled to $[0, 1]$. All simulations incorporating a μCHP also encompass the simulation of the respective household that is heated by this μCHP. This implies a simulation of the respective heat demand, heat use, different weather conditions or heat losses by thermal diffusion processes.

For our simulations, we used the model of a modulating μCHP-plant with the following specification:

- Minimum electrical power: 1.3 kW,
- Maximum electrical power: 4.7 kW,
- Minimum thermal power: 4 kW,
- Maximum thermal power: 12.5 kW,
- After shut down, a device has to stay off for at least 2 h.

A modulating CHP is a generator that may vary the level of electrical power output within a certain range resulting in different thermal power output respectively. The relationship between electrical (active) power and thermal power was modeled after Fig. 6. In order to gain more degrees of freedom for varying active power, each CHP is equipped with an 800 ℓ thermal buffer store. Thermal energy consumption is simulated by a model of a detached house with its several heat losses (heater is supposed to keep the indoor temperature on a constant level) and randomized warm water drawing for gaining more diversity among the devices.

For each simulated household, we implemented an agent capable of simulating the CHP (and surroundings and auxiliary devices) on a meso-scale level with energy flows among different model parts but with no technical details. All implementations have been done using the Java programming language. For the implementation of the multi-agent system prototype we used the MASON multi-agent simulator toolkit [26]. The support vector algorithm has been implemented using an adaption of the sequential minimization technique from [37].

All simulations have so far been done with a time resolution of 15 minutes for different forecast horizons. For each simulation, we have run 200 test series with each CHP randomly initialized with different buffer charging levels, temperatures and water drawing profiles. We tested schedules with dimension 8, 16, 32, 48, 64, and 96 periods with groups of 5, 10, 30, 100, 500, 750, and 1000 CHP. For each simulation, we have chosen some random target schedule on a

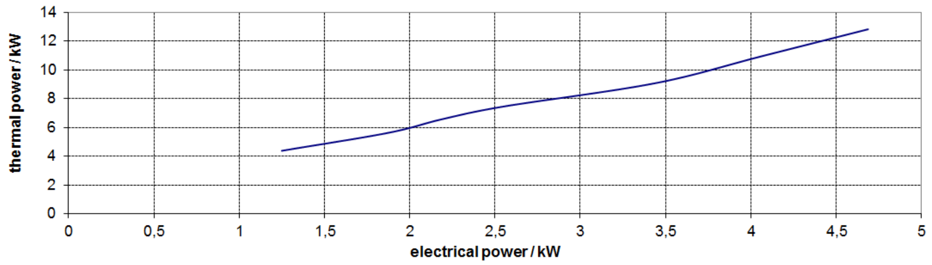


Fig. 6. Relationship between electrical and thermal power for an EcoPower CHP; modified after test bench runs from [45].

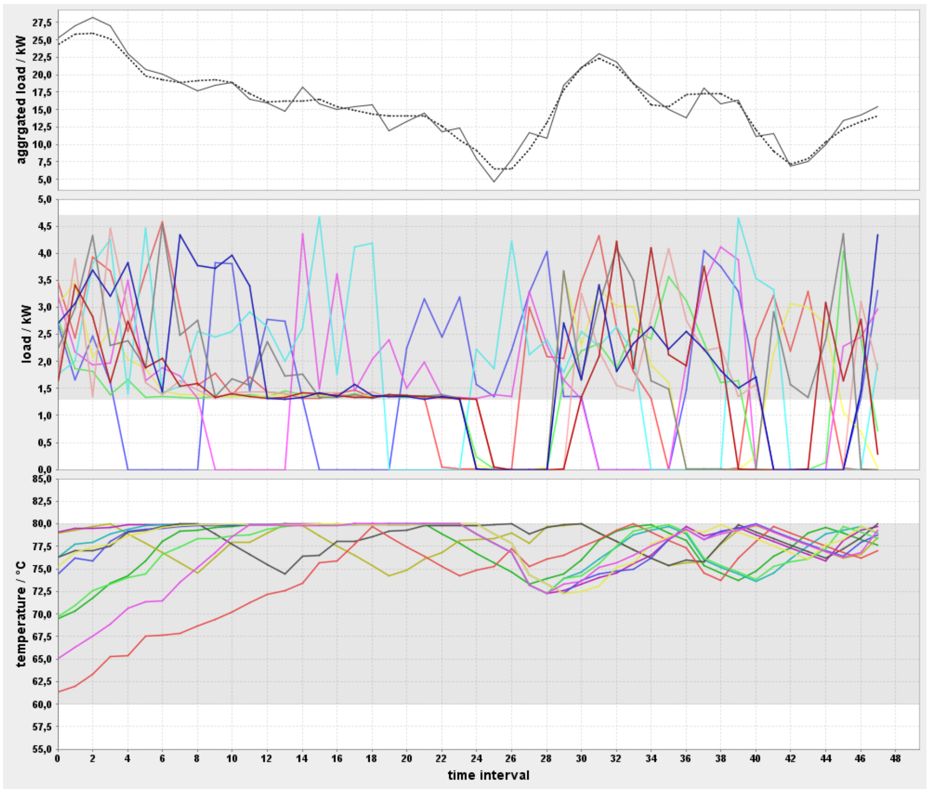


Fig. 7. Optimization result for a winter day scenario with 10 CHP (EcoPower with randomly initialized storage charging) for a time horizon of 48 15-minute intervals.

rather high level. This leads to a rather high charging of the thermal buffer stores on the long run but on the other hand forces the units to go to their limit and demonstrates that the method copes well with the buffer constraints as well.

Figure 7 shows a result (solid line in the top chart) for a group of 10 CHP that try to reach a given objective schedule (dashed line). The resulting schedules for each single CHP are depicted in the middle chart with the allowed active power band for modulation highlighted in grey. The bottom chart shows the temperatures in the thermal buffer store resulting from operating the respective electrical schedules; again with the allowed range highlighted in grey.

The desired objective schedule has been randomly chosen. These schedules have been generated in a way that they are of a reasonable magnitude order according to the capabilities of the optimized CHP, but without any guarantee that a perfect adaption might be achievable.

Figure 8 shows a similar simulation run, but for a time horizon of a whole day. Fig. 9 shows the result for optimizing a larger bunch of 30 CHP. As might have been expected, the result schedule gets closer to the target schedule if

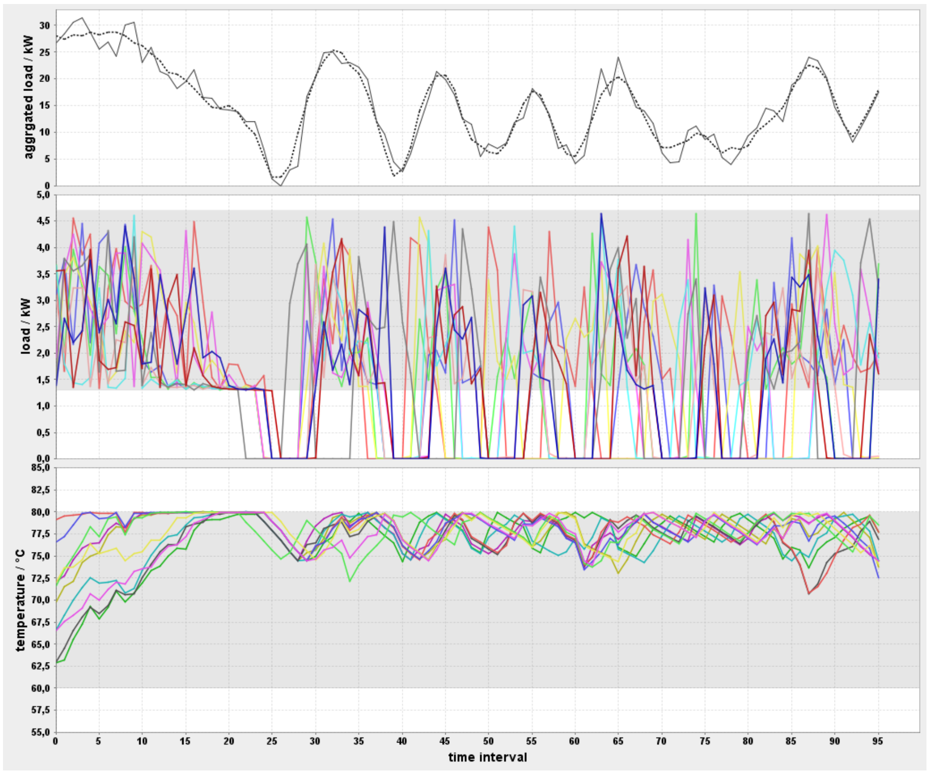


Fig. 8. Optimization result for a spring day scenario with 10 CHP (EcoPower with randomly initialized storage charging) for a time horizon of a whole day in 15-minute intervals.

more CHP are involved. This is mainly due to the availability of more degrees of freedom for the system as a whole.

As a next step, we scrutinized the speed of convergence and convergence behaviour of our algorithm. Figure 10 shows the result of some measuring series. It is noticeable that the fitness (the difference between aggregated and target solution) almost decreases strictly notwithstanding the uncoordinated, heuristic character of the approach. If the algorithm is conducted asynchronously, there are indeed more fluctuations that lead to temporarily degradations of the fitness. This can be easily overcome if the agent in charge the solution rejects solution updates that lead to degradation.

The number of necessary iterations is acceptably small, what can also be seen in Table 1, where some mean CPU time results (Java implementation on Core 2, 3 GHz) for different scenarios are listed. The simulation time t_{sim} reflects the time that is necessary for the whole simulation including the preceding calculations of the set of feasible schedules for each agent, for the calculation of all support vector models and all mapping functions γ on a single machine.

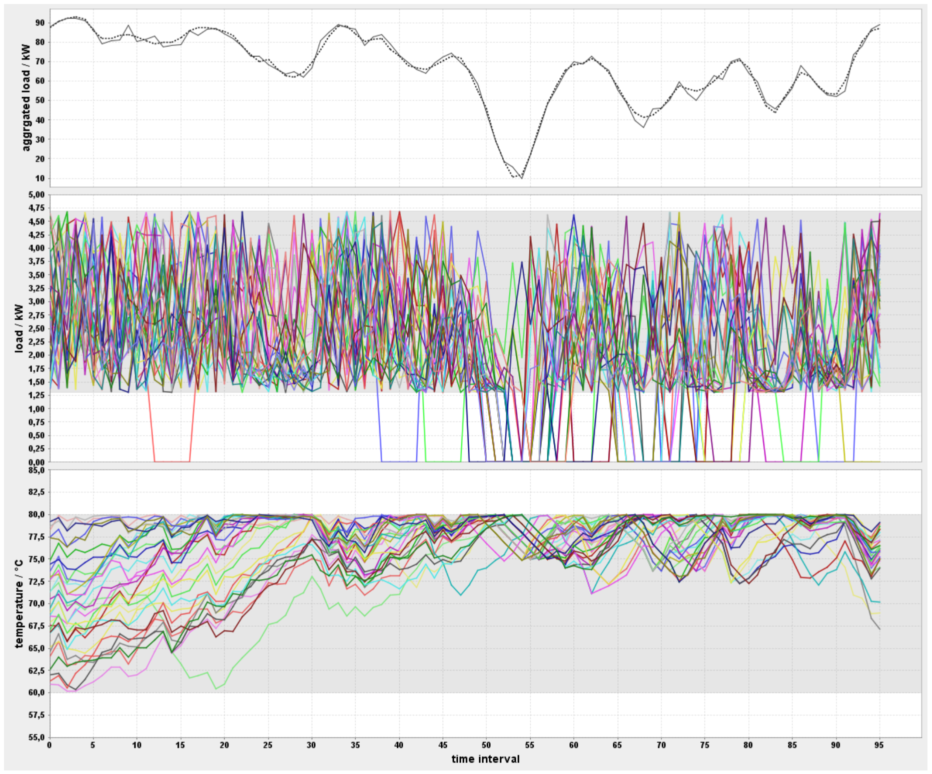
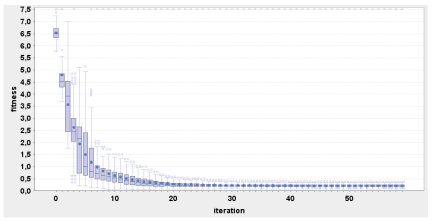


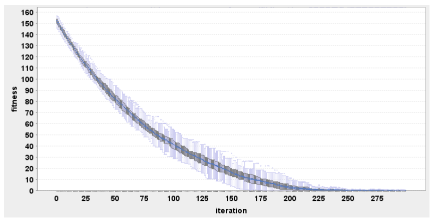
Fig. 9. Optimization result for a scenario with 30 CHP for 96 15-minute intervals. This amounts to a 2880-dimensional search space.

In a distributed productive system these calculations would be done in parallel. Considering the additional complexity that is entailed on solution evaluation, the number of support vectors that make up a model is decisive. Step 1 of calculating the mapping grows quadratically with the number of support vectors (matrix-vector multiplication). Additionally, the number of iterations necessary for finding the pre-image in step 3 has to be considered. Empirically, during our experiments, we observed for instance a mean number of 36.3 ± 26.4 for the case of 8-dimensional schedules to reach convergence with 10^{-6} accuracy.

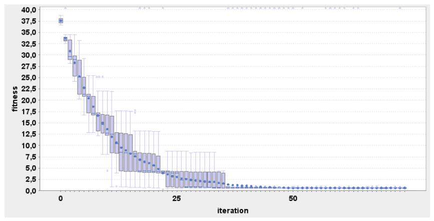
The time necessary for the mere optimization is comparably small. In order to be able to simulate larger scenarios, we are currently thinking of distributing the simulation, too. So far, we tested a scenario with a group of 50.000 CHP. We parallelized all calculations (simulation of each CHP and optimization of the whole group) on a system with Core i3 processor and were able to complete all calculations (1 day planning horizon) within about 18 minutes. The optimization resulted in a residual error of only about 0.8 percent (compared with the worst case operation that the units could do).



(a)



(b)



(c)

Fig. 10. Convergence for different scenarios: 10(a): 5 CHP and, 8 periods; 10(b): 100 CHP, 8 periods, 10(c): 10 CHP, 96 periods.

Table 1. CPU time for algorithm and simulation regarding different problem sizes. Quality as mean euclidean distance in kW for n_A agents, k iterations and schedules of d intervals of 15 min.

d	n_A	k	t_{sim} / s	t_{opt} / s	QUALITY
8	10	75	4.71 $\pm$ 0.23	0.006$\pm$0.008	0.054$\pm$0.023
8	100	750	45.2 $\pm$ 0.74	0.061$\pm$0.009	0.045$\pm$0.02
32	100	250	382.59 $\pm$ 27.24	0.049$\pm$0.008	1.05$\pm$1.09
96	10	750	251.4 $\pm$ 4.5	0.498$\pm$0.127	0.049$\pm$0.08

Comparing the synchronously (randomized round robin) operated and the fully parallel, asynchronously operated variant of the optimization part, figure 11 shows the convergence behaviour of both approaches by example. Depicted are the results of 500 runs each (the darker the color the more results). The iteration number on the x-axis denotes the number of solution update. Due to the inherent stochasticity of the scenario (250 CHP, 8-dimensional schedule), all errors have been scaled by the individual initial error of the randomly instantiated problem in order to make them comparable. Although the synchronous approach (bottom chart) performs slightly better in term of necessary solution updates, the asynchronous approach takes advantage in (and thus overcompensates this disadvantage by) parallel calculations of the mapping and should thus be preferred.

Finally, figure 12 shows a result from a larger mixed scenario with two different types (in power magnitude) of CHP. Having different DERs in a scenario,

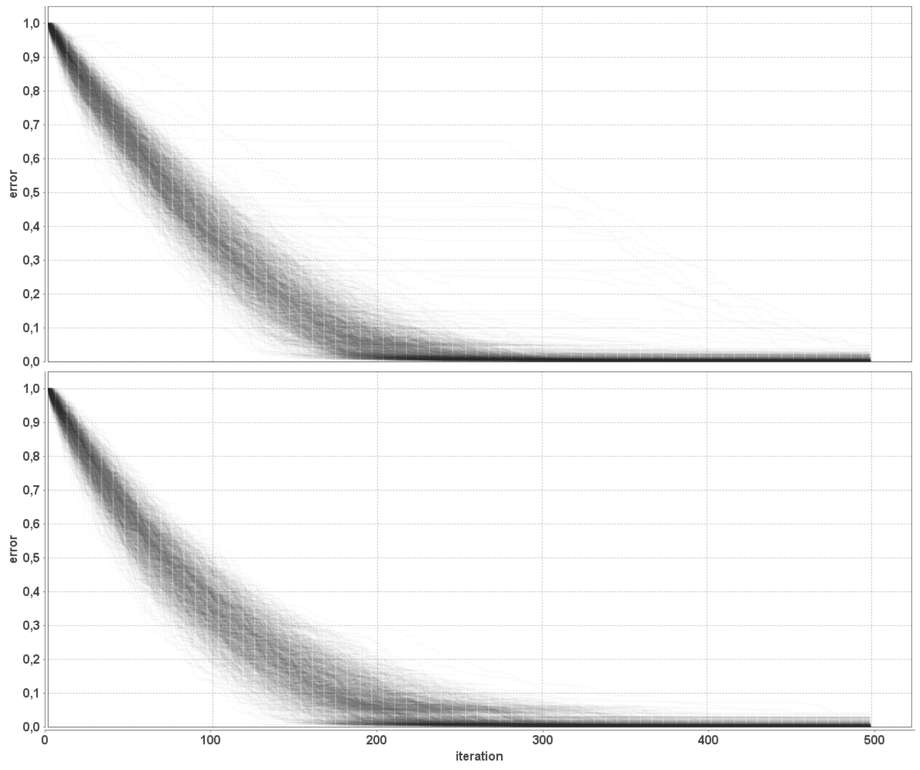


Fig. 11. Comparison of the convergence of synchronous (bottom) and asynchronous (top) operation of the greedy algorithm.

often leads to a better adaption to the target schedule as has also been seen in similar scenarios with a mixture of CHP and refrigerators e.g. in [7].

Another important issue is the question for the size of the search space model that almost exclusively depends on the number of supporting schedules. Figure 13 shows one example with the number of supporting schedules as a function of the time horizon. This example shows that events like increased heat demand in the morning (showering) or in the evening (heater) leads to an escalating increase of information and therefore to a respectively increased number of supporting schedules for appropriate encoding. So far, we have observed that the size of necessary information in fact is a trade-off between accuracy, size and calculation complexity and hence a matter of finding the optimal parameters.

A further issue is the consideration of possible errors that might occur in encoding. It may happen that the enclosing contour adapts not good enough to the point cloud [7]. This may additionally cause two (or more) subregions to be misleadingly considered as one connected region, although they are actually topologically separated by regions of invalid schedules. In both cases, the

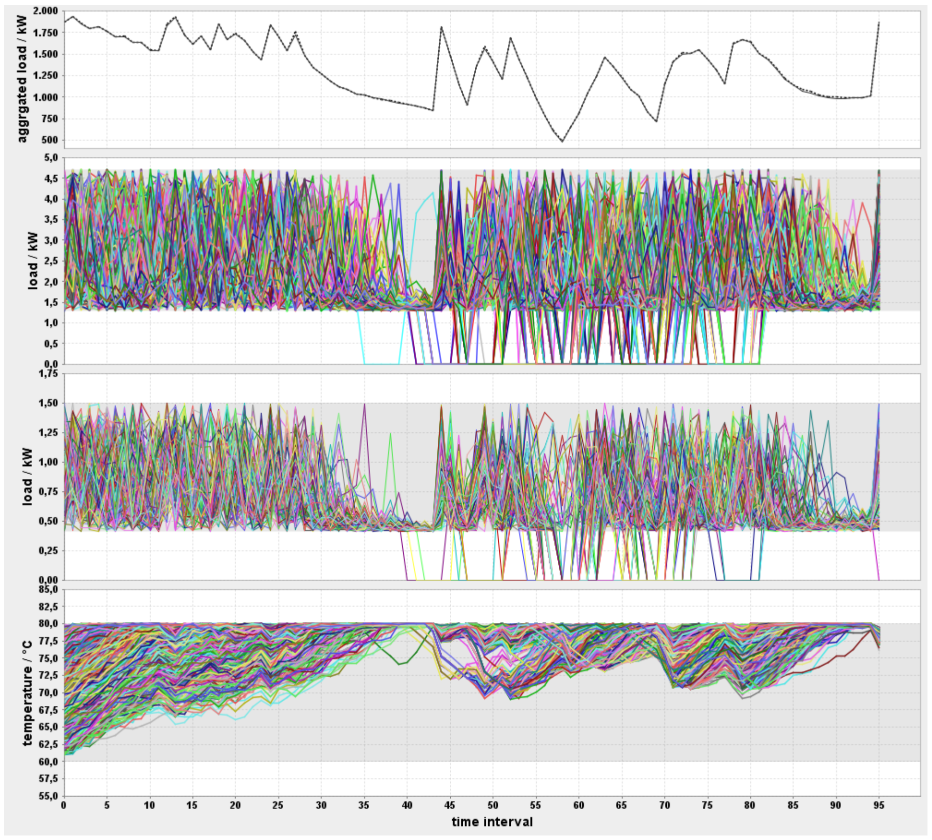


Fig. 12. Optimization result for a scenario with 750 CHP for 96 15-minute intervals. This amounts to a 72000-dimensional search space.

search space model would reconstruct a too large feasible region and consider some schedules incorrectly as operable. Whether this happens and how large these errors are, depends on the choice of parameters.

It is mainly parameter σ in (4), the width of the Gaussian kernel, that determines how smooth the resulting boundary contour adapts to the data. As the boundary adapts closer to the data with a shrinking value for σ, the enclosed region starts separating into separate regions. A smaller value for σ thus leads to a more precise description but on the other hand also to a higher number of support vectors needed for description.

This effect might partly be overcome by fitting parameters of the energy unit simulation model such that the feasible region of alternative schedules is determined too narrow in advance in order to compensate for a too wide description. On the other hand this fact might lead to a too narrow description that misses some schedules near the boundary.

Within the scenarios scrutinized here, most of the encoded schedules are at least partially based on forecasts, i.e. on the anticipated heat demand, weather conditions or similar, so that uncertainty is already inherent. If the error, that is additionally induced by our method is small compared to the already prevalent one, it can be neglected during the optimization process. Error minimization can be achieved by parameter selection.

In any case, when a certain schedule has been chosen by the planning algorithm, it has to be validated by the DER or respectively by the agent with the help of the simulation model before actually operating it. Conditionally, it has be mapped to the nearest valid schedule, where required.

Hence, a crucial point and a remaining open issue is the parameterization of the method. For the previously described simulations, we experimentally determined the best parameters. For this reason, one of our next steps will be to develop a tuning algorithm that is able to (at least semi-) automatically derive optimal parameters (sample size, kernel parameters, SMO parameters, etc.) for an optimal encoding in the sense of the best trade-off between accuracy and number of supporting schedules.

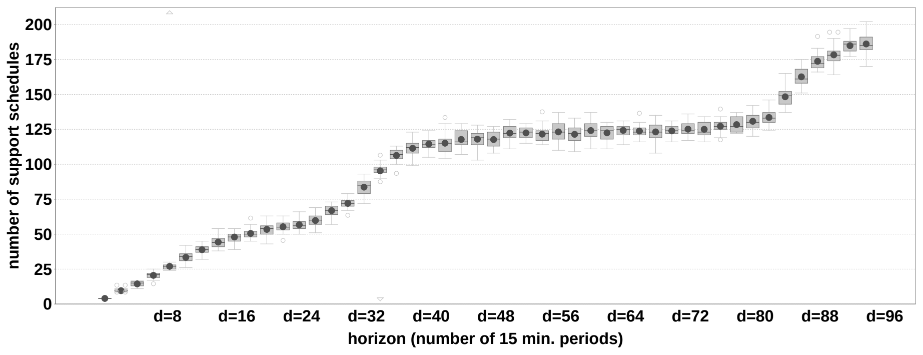


Fig. 13. Number of supporting schedules as a function of the horizon, i.e. the data dimension.

5. Conclusion and further work

We have presented a new approach for distributed optimization and control of distributed energy resources for smart grid scenarios with a large number of controllable entities. This approach is based on two new methodologies:

- A model-based strategy for handling constraints in distributed optimization scenarios that may also be used for finding nearby feasible solutions by harnessing a learned model of the feasible region.
- A well scaling greedy algorithm for harnessing that strategy during the search for an optimal partition of the requested schedule for different DERs.

We have demonstrated that the greedy heuristics scales well with the number of participating devices because the most expensive calculations may be done in parallel in advance by each controllable device.

This work is part of the ongoing smart grid research association Smart Nord (`http://smartnord.de`). This method is currently integrated into a simulation environment for large smart grid scenarios with up to 50.000 electrical units. One goal is to develop an integrated simulation environment based on the mosaik framework [40] for scrutinizing scenarios for market-based planning of active power provision by self-organized coalitions with optimization and re-scheduling capabilities [33]. At the same time ancillary services (e.g. for frequency or voltage stability) are integrated.

Clearly, building such large and diverse scenarios involves the integration of many more types of distributed energy resources. So far, we are planning to integrate among others models for photovoltaic panels (limited controllability), heat pumps, co-generation, night storage heater, white goods (fridge, dishwasher, etc.), batteries, air condition as well as non controllable resources like wind energy.

From this point of view it becomes clear that a model-based approach for the integration of all these different energy units (and the integration of future, yet unknown ones) is indispensable. So far, we paved the way from the automatic conversion of the scope of action of an energy unit or its simulation model to a standard search space model that can be easily integrated into planning and optimization algorithms. In this way, it now becomes easy to commonly integrate a diverse set of different models jointly into distributed algorithms.

Due to this abstraction by search space model and mapping, all energy units become indistinguishable for algorithms and may thus be accessed always in the same, standardized way.

Acknowledgments. The Lower Saxony research network 'Smart Nord' acknowledges the support of the Lower Saxony Ministry of Science and Culture through the Niedersächsisches Vorab grant programme (grant ZN 2764).

References

1. Bary, C.: Coincidence-factor reationships of electric-service-load characteristics. American Institute of Electrical Engineers, Transactions of the 64(9), 623–629 (1945)
2. Beer, S., Appelrath, H.J., Sonnenschein, M.: Towards a self–organization mechanism for agent associations in electricity spot markets. In: Informatik 2011 - Workshop IT für die Energiesysteme der Zukunft (10 2011)
3. Ben-Hur, A., Siegelmann, H.T., Horn, D., Vapnik, V.: Support vector clustering. Journal of Machine Learning Research 2, 125–137 (2001)
4. Blank, M., Gerwinn, S., Krause, O., Lehnhoff, S.: Support vector machines for an efficient representation of voltage band constraints. In: Innovative Smart Grid Technologies. IEEE PES (2011)

5. Born, F.: Aiding Renewable Energy Integration Through Complimentary Demand-supply Matching. University of Strathclyde (2001), http://books.google.de/books?id=w0BCHQAACAAJ
6. Bremer, J.: Ontology based description of der's learned environmental performance indicators. In: Donnellan, B., Lopes, J.P., Martins, J., Filipe, J. (eds.) Proceedings of the 1st International Conference on Smart Grids and Green IT Systems – Smart-Greens 2012. pp. 107–112. SciTePress, Porto, Portugal (04 2012)
7. Bremer, J., Rapp, B., Sonnenschein, M.: Encoding distributed search spaces for virtual power plants. In: IEEE Symposium Series in Computational Intelligence 2011 (SSCI 2011). Paris, France (4 2011)
8. Bremer, J., Rapp, B., Sonnenschein, M.: Including Environmental Performance Indicators into Kernel based Search Space Representations. Information Technologies in Environmental Engineering (ITEE 2011) (2011)
9. Bremer, J., Sonnenschein, M.: A distributed greedy algorithm for constraint-based scheduling of energy resources. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (eds.) Federated Conference on Computer Science and Information Systems - Fed-CSIS 2012, Wroclaw, Poland, 9-12 September 2012, Proceedings. pp. 1285–1292 (2012)
10. Bremer, J., Rapp, B., Sonnenschein, M.: Support vector based encoding of distributed energy resources' feasible load spaces. In: IEEE PES Conference on Innovative Smart Grid Technologies Europe. Chalmers Lindholmen, Gothenburg, Sweden (2010)
11. Brusan, A.: Very short literature survey from supervised learning to surrogate modeling. CoRR abs/1203.4788 (2012)
12. Chu, C.S., Tsang, I.W., Kwok, J.T.: Scaling up support vector data description by using core-sets. In: Proceedings of 2004 IEEE International Joint Conference on Neural Networks. vol. 1, pp. 430–435 (2004)
13. Evangelista, P., Embrechts, M., Szymanski, B.: Taming the curse of dimensionality in kernels and novelty detection. In: Abraham, A., de Baets, B., Kppen, M., Nickolay, B. (eds.) Applied Soft Computing Technologies: The Challenge of Complexity, Advances in Soft Computing, vol. 34, pp. 425–438. Springer Berlin Heidelberg (2006), http://dx.doi.org/10.1007/3-540-31662-0_33
14. Franch, T., Scheidt, M., Stock, G.: Current and future challenges for production planning systems. In: Kallrath, J., Pardalos, P.M., Rebennack, S., Scheidt, M., Pardalos, P.M. (eds.) Optimization in the Energy Industry, pp. 5–17. Energy Systems, Springer Berlin Heidelberg (2009)
15. Gano, S.E., Kim, H., Brown II, D.E.: Comparison of three surrogate modeling techniques: Datascape, kriging, and second order regression. In: Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7048. Portsmouth, Virginia (2006)
16. Gatterbauer, W.: Economic efficiency of decentralized unit commitment from a generator's perspective. In: Ilic, M. (ed.) Engineering Electricity Services of the Future. Springer (2010)
17. Guan, X., Zhai, Q., Papalexopoulos, A.: Optimization based methods for unit commitment: Lagrangian relaxation versus general mixed integer programming. vol. 2, p. 1100 Vol. 2 (2003), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1270468
18. Hinrichs, C., Lehnhoff, S., Sonnenschein, M.: A Decentralized Heuristic for Multiple-Choice Combinatorial Optimization Problems. In: Operations Research 2012 – Selected Papers of the International Conference on Operations Research (OR

2012). Springer, Hannover, Germany (2013), http://www-ui.informatik.uni-oldenburg.de/download/Publikationen/HLS12.pdf
19. Juszczak, P., Tax, D., Duin, R.P.W.: Feature scaling in support vector data description. In: Deprettere, E., Belloum, A., Heijnsdijk, J., van der Stappen, F. (eds.) Proc. ASCI 2002, 8th Annual Conf. of the Advanced School for Computing and Imaging. pp. 95–102 (2002)
20. Kamper, A., Esser, A.: Strategies for decentralised balancing power. In: A. Lewis, S. Mostaghim, M.R. (ed.) Biologically-inspired Optimisation Methods: Parallel Algorithms, Systems and Applications, pp. 261–289. No. 210 in Studies in Computational Intelligence, Springer, Berlin, Heidelberg (Juni 2009), http://dx.doi.org/10.1007/978-3-642-01262-4
21. Kamphuis, R., Warmer, C., Hommelberg, M., Kok, K.: Massive coordination of dispersed generation using powermatcher based software agents. In: 19th International Conference on Electricity Distribution (May 2007)
22. Kok, K., Derzsi, Z., Gordijn, J., Hommelberg, M., Warmer, C., Kamphuis, R., Akkermans, H.: Agent-based electricity balancing with distributed energy resources, a multiperspective case study. Hawaii International Conference on System Sciences 0, 173 (2008)
23. Kramer, O.: A review of constraint-handling techniques for evolution strategies. Appl. Comp. Intell. Soft Comput. 2010, 1–19 (January 2010)
24. Kwok, J., Tsang, I.: The pre-image problem in kernel methods. Neural Networks, IEEE Transactions on 15(6), 1517–1525 (2004)
25. Laskov, P., Gehl, C., Krüger, S., Müller, K.: Incremental support vector learning: Analysis, implementation and applications. Journal of Machine Learning Research 7, 1906–1936 (2006)
26. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: Mason: A multiagent simulation environment. Simulation 81(7), 517–527 (Jul 2005), http://dx.doi.org/10.1177/0037549705058073
27. Lukovic, S., Kaitovic, I., Mura, M., Bondi, U.: Virtual power plant as a bridge between distributed energy resources and smart grid. Hawaii International Conference on System Sciences 0, 1–8 (2010)
28. Mao, Y., Li, M.: Optimal reactive power planning based on simulated annealing particle swarm algorithm considering static voltage stability. In: Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation - Volume 01. pp. 106–110. ICICTA '08, IEEE Computer Society, Washington, DC, USA (2008), http://dx.doi.org/10.1109/ICICTA.2008.427
29. Mihailescu, R.C., Vasirani, M., Ossowski, S.: Dynamic coalition adaptation for efficient agent-based virtual power plants. In: Proceedings of the 9th German conference on Multiagent system technologies. pp. 101–112. MATES'11, Springer-Verlag, Berlin, Heidelberg (2011)
30. Mika, S., Schölkopf, B., Smola, A., Müller, K.R., Scholz, M., Rätsch, G.: Kernel PCA and de-noising in feature spaces. In: Proceedings of the 1998 conference on Advances in neural information processing systems II. pp. 536–542. MIT Press, Cambridge, MA, USA (1999)
31. Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product in Optimization Using Designed Experiments. John Wiley & Sons, Inc., New York, NY, USA, 1st edn. (1995)
32. Neddermeijer, H.G., van Oortmarssen, G.J., Piersma, N., Dekker, R.: A framework for response surface methodology for simulation optimization. In: Proceedings of the 32nd conference on Winter simulation. pp. 129–136. WSC '00, So-

ciety for Computer Simulation International, San Diego, CA, USA (2000), `http://dl.acm.org/citation.cfm?id=510378.510401`

33. Nieße, A., Lehnhoff, S., Tröschel, M., Uslar, M., Wissing, C., Appelrath, H.J., Sonnenschein, M.: Market–based self–organized provision of active power and ancillary services. IEEE (06 2012)
34. Nieße, A., Lehnhoff, S., Tröschel, M., Uslar, M., Wissing, C., Appelrath, H.J., Sonnenschein, M.: Market-based self-organized provision of active power and ancillary services: An agent-based approach for smart distribution grids. In: COMPENG. pp. 1–5. IEEE (2012)
35. Parzen, E.: On estimation of a probability density function and mode. The Annals of Mathematical Statistics 33(3), pp. 1065–1076 (1962), `http://www.jstor.org/stable/2237880`
36. Pereira, J., Viana, A., Lucus, B., Matos, M.: A meta-heuristic approach to the unit commitment problem under network constraints. International Journal of Energy Sector Management 2(3), 449–467 (2008)
37. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods. pp. 185–208. MIT press (1999)
38. Ramchurn, S.D., Vytelingum, P., Rogers, A., Jennings, N.R.: Agent-based control for decentralised demand side management in the smart grid. In: Sonenberg, L., Stone, P., Tumer, K., Yolum, P. (eds.) AAMAS. pp. 5–12. IFAAMAS (2011)
39. Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K.R., Rätsch, G., Smola, A.: Input space vs. feature space in kernel-based methods. IEEE Transactions on Neural Networks 10(5), 1000–1017 (1999)
40. Schütte, Steffen; Scherfke, S.S.M.: mosaik – smart grid simulation api. In: Donnellan, B., Lopes, J.P., Martins, J., Filipe, J. (eds.) Proceedings of the 1st International Conference on Smart Grids and Green IT Systems – SmartGreens 2012. SciTePress, Porto, Portugal (04 2012)
41. Sonnenschein, M., Appelrath, H.J., Hofmann, L., Kurrat, M., Lehnhoff, S., Mayer, C., Mertens, A., Uslar, M., Nieße, A., Tröschel, M.: Dezentrale und selbstorganisierte koordination in smart grids. In: VDE-Kongress 2012 Smart Grid Intelligente Energieversorgung der Zukunft. VDE (11 2012)
42. Tavakkoli, A., Nicolescu, M., Nicolescu, M., Bebis, G.: Incremental svdd training: Improving efficiency of background modeling in videos. In: Cristea, P. (ed.) Signal and Image Processing. Acta Press, Calgary, Canada (2008)
43. Tax, D.M.J., Duin, R.P.W.: Data domain description using support vectors. In: ESANN. pp. 251–256 (1999)
44. Tax, D.M.J., Duin, R.P.W.: Support vector data description. Mach. Learn. 54(1), 45–66 (2004)
45. Thomas, B.: Mini-Blockheizkraftwerke: Grundlagen, Gerätetechnik, Betriebsdaten. Vogel Buchverlag (2007)
46. Tröschel, M., Appelrath, H.J.: Towards reactive scheduling for large-scale virtual power plants. In: Braubach, L., van der Hoek, W., Petta, P., Pokahr, A. (eds.) MATES. Lecture Notes in Computer Science, vol. 5774, pp. 141–152. Springer (Sep 2009)
47. Xiong, W., Li, M.j., Cheng, Y.l.: An improved particle swarm optimization algorithm for unit commitment. In: Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation - Volume 01. pp. 21–25. ICICTA '08, IEEE Computer Society, Washington, DC, USA (2008), `http://dx.doi.org/10.1109/ICICTA.2008.363`

Jörg Bremer is research assistant at the Department for Computing Science at the Carl von Ossietzky University Oldenburg, Germany. He recieved a Diploma (Environmental Informatics) in the field of agent based simulations of household energy consumption in decentralized scenarios (2006). He has been working on several projects on energy management and computational intelligence in smart grids. In addition, he has been working as a research assistant at the Department for Business Information Systems and at the OFFIS Institute for Information Technology in Oldenburg. He also teaches at the University of Oldenburg in the field of decentralized energy systems. Currently, Mr Bremer is a PhD Student at the chair of Prof. Sonnenschein.

Michael Sonnenschein is professor of Computer Science at the Carl von Ossietzky University of Oldenburg, Germany. He studied Computer Science and Mathematics at the Aachen University of Technology (Diploma 1979); PhD in computer 1983, Habilitation in Computer Science 1991, both at Aachen University of Technology. Since 1991 he is professor for Computer Science at Oldenburg University. As a member of the executive board energy of the OFFIS Institute for Information Technology he headed several projects on energy management in smart grids. His research interests include techniques for modelling, simulation, and heuristic optimization in environmental applications, particularly in smart grids.

Received: March 4, 2013; Accepted: August 3, 2013.